Chương 5

THIẾT KẾ CÁC CHƯƠNG TRÌNH SONG SONG

5.1. SONG SONG THỦ CÔNG VÀ TỰ ĐỘNG SONG SONG

Khi thiết kế và phát triển các chương trình song song sẽ cho một quá trình rất thủ công. Đó là lập trình viên thường phải chịu trách nhiệm cho cả hai việc:

- Nhận dạng các phần công việc có thể thực thi song song và

- Thực hiện việc song song hóa.

Khi viết code cho các chương trình song song bao giờ cũng tiêu tốn nhiều thời gian, phức tạp, dễ bị lỗi và xử lý lặp.

Những năm gần đây, đã có các công cụ khác nhau để hỗ trợ các lập trình viên chuyển đổi chương trình tuần tự sang các chương trình song song - nhằm giảm tối đa phần thủ công đã nêu trên. Các loại công cụ phổ biến nhất được sử dụng để tự động song song hóa một chương trình tuần tự là trình biên dịch song song hóa hoặc bộ tiền vi xử lý.

Thủ công

Nếu việc song song hóa chỉ hoàn toàn do lập trình viên làm, được gọi là song song hóa thủ công toàn phần. Nếu việc song song hóa có hỗ trợ một phần của lập trình viên, được gọi là song song hóa thủ công bán phần.

Tự động

Một trình biên dịch song song hóa nói chung làm việc theo hai cách khác nhau:

Hoàn toàn tự động. Trình biên dịch phân tích mã nguồn và

nhận dạng cơ hội (các vùng) có thể xử lý song song.

Phân tích khả năng tính toán song song. Bao gồm việc xác định có cách nào để xử lý song song hay không, vì có thể sẽ hoặc có chi phí cao hoặc sẽ cải thiện được hiệu quả rất lớn. Các vòng lặp (Do, For,...) là những mục tiêu hay dùng nhất cho việc song song hóa tự động.

Bán tự động

Có sự can thiệp của lập trình viên như:

- Các chỉ thị của lập trình viên như thế nào và cần chèn vào vị trí nào.

- Khi sử dụng các chỉ thị của trình biên dịch hoặc có thể là cờ biên dịch, lập trình viên phải chỉ rõ cho trình biên dịch cách thức để mã hóa song song.

Nhận xét

Nếu chúng ta đang bắt đầu với một mã tuần tự hiện có và có sự ràng buộc về thời gian hoặc ngân sách, thì tự động song song hóa có thể là câu trả lời tốt nhất. Tuy nhiên, có một số điều quan trọng là hãy áp dụng thận trọng tự động hóa song song, vì rằng:

- Các kết quả sai có thể sinh ra.

- Hiệu năng thực sự có thể bị giảm đi.

- Có thể kém linh hoạt hơn nhiều so với song song thủ công.

- Bị giới hạn trong một tập con (chủ yếu là các vòng lặp) của code.

- Có thể không thực sự cần viết mã song song nếu khi phân tích cho thấy có sự cưỡng bức hoặc viết mã quá phức tạp.

5.2. NHẬN THỨC VỀ VẤN ĐỀ VÀ CHƯƠNG TRÌNH CÓ THỂ SONG SONG HÓA

Bước đầu tiên trong việc phát triển phần mềm song song là

hiểu được vấn đề - hiểu được công việc chúng ta cần giải quyết song song. Nếu chúng ta đang bắt đầu với một chương trình tuần tự, điều này cũng đòi hỏi sự hiểu biết thuật toán, code và cả ngôn ngữ lập trình của chương trình tuần tự đó.

Trước khi dùng thời gian và sức lực nhằm phát triển giải pháp song song cho một vấn đề, hãy xác định có phải đó là một trong những vấn đề mà trên thực tế có thể song song hóa được hay không.

Ví dụ về vấn đề có thể song song hóa được (Parallelizable):

Tính toán năng lượng tiềm năng cho mỗi trong vài nghìn phân hủy độc lập của một phân tử. Khi thực hiện, tìm năng lượng phân hủy tối thiểu. Vấn đề này có thể được giải quyết song song. Mỗi một phân hủy của phân tử có thể xác định một cách độc lập. Tính toán năng lượng phân hủy tối thiểu cũng là một vấn đề có thể song song hóa.

Hoặc khi tính tổng của hai véctơ A(n) và B(n) cùng có n phần tử, việc cộng hai phần tử nào đó là độc lập với các phần tử còn lại. Vấn đề này cũng có thể được giải quyết song song.

Ví dụ về vấn đề không song song hóa được (non- parallelizable):

Tính chuỗi Fibonacci (1, 1, 2, 3, 5, 8, 13, 21,...) bằng cách sử dụng công thức:

$$F(k+2) = F(k+1) + F(k)$$

Đây là bài toán không song song hóa được bởi vì tính số hạng của dãy Fibonacci theo công thức là phụ thuộc chứ không phải là độc lập. Việc tính giá trị thứ k + 2 phải sử dụng những giá trị của cả hai k + 1 và k. Ba điều kiện không thể được tính toán một cách độc lập và do đó, không thể tính toán song song.

Hoặc khi tính n! trong đó n N*, thì dùng chu trình For đều không song song hóa được.

```
GiaiThua = 1
```

```
For i = 1 to n
 GiaiThua * = i
Next i
```

Hoặc dùng hàm đệ quy:

*GiaiThua(n) = n*GiaiThua(n-1)*

Xác định các (điểm nóng) vị trí trong chương trình có thể/không tính toán song song được.

Cần phải biết vị trí trong chương trình, nơi mà hầu hết các tác vụ thực tế đang được thực hiện. Đa số các chương trình khoa học và kỹ thuật thường thực hiện hầu hết công việc của mình ở một vài nơi - gọi là điểm nóng của chương trình. Chính những điểm nóng mới có nhiều khả năng song song hóa được. Như vậy, nên tập trung phân tích và thiết kế song song hóa vào các điểm nóng và bỏ qua những phần của chương trình cho là ít sử dụng CPU.

Nhận diện tắc nghẽn trong chương trình

Nơi tắc nghẽn thường xảy ra thường là

- Có các khu vực chậm không cân đối, hoặc là do làm công việc song song nên tạm dừng hoặc được hoàn trả chậm? Ví dụ, I/O thường làm chậm việc tải một chương trình.

- Có thể có khả năng cơ cấu lại các chương trình hay sử dụng một thuật toán khác nhau để giảm hoặc loại bỏ các khu vực chậm nhưng không cần thiết.

Nhận diện các hạn chế xử lý song song

- Lớp hạn chế phổ biến nhất là sự phụ thuộc dữ liệu, như đã chỉ mục “Sự phụ thuộc dữ liệu” (Cụ thể là việc tính các số hạng của dãy Fibonacci hoặc tính n! đã nêu ở trên).

- Nghiên cứu các thuật toán khác nếu có thể để thay cho thuật toán hiện thời. Điều này có thể là rất quan trọng khi thiết kế

một ứng dụng song song.

5.3. PHƯƠNG PHÁP PHÂN HOẠCH DỮ LIỆU VÀ PHÂN CHIA CHƯƠNG TRÌNH ĐỂ SONG SONG HÓA

Một trong những bước đầu tiên trong việc thiết kế một chương trình song song là chia vấn đề (bài toán) thành các "khối" con rời rạc mà chúng có thể được phân tán thành nhiều tác vụ, sau đó mỗi tác vụ được phân phối cho các CPU để thực hiện. Điều này được gọi là phân vùng hay phân hoạch.

Có hai cách cơ bản để phân hoạch tính toán giữa các tác vụ song song: phân hoạch theo miền dữ liệu và phân hoạch theo chức năng tính toán.

5.3.1. Phân hoạch theo miền dữ liệu (Domain Decomposition)

Phân hoạch kiểu này, các dữ liệu liên kết với một vấn đề được phân chia ra thành các phần độc lập. Sau đó, mỗi tác vụ song song hoạt động trên một phần của dữ liệu đã được phân hoạch. Ví dụ tính tổng hai véctơ C = A + B, hình 5.1.

A(1)	A(2)	A(3)	A(4)	....	A(i)	A(i+1)	A(i+2)	...	A(n)
B(1)	B(2)	B(3)	B(4)	....	B(i)	B(i+1)	B(i+2)	...	B(n)

Hình 5.1. Các phần tử của các mảng A và B

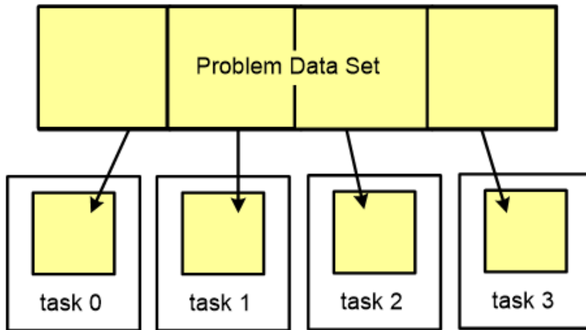
Được chia thành các khối sau để đưa cho các tác vụ thực hiện phép cộng, hình 5.2.

A(1)	A(2)	A(3)	A(4)	...	A(i)	A(i+1)	A(i+2)	...	A(n-2)	A(n-1)	A(n)
B(1)	B(2)	B(3)	B(4)	...	B(i)	B(i+1)	B(i+2)	...	B(n-2)	B(n-1)	B(n)
Task 0 (Tác		Task i (Tác		Task k (Tác							

vụ 0)	vụ i)	vụ k)

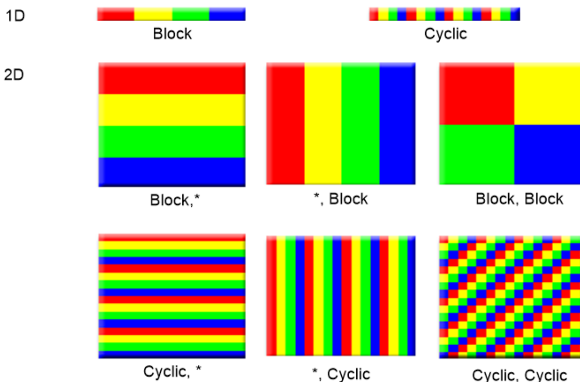
Hình 5.2. Các tác vụ cộng các phần tử của hai mảng

Có nhiều cách khác nhau thực hiện các tác vụ trên các vùng dữ liệu, ví dụ theo block - khối, nghĩa là mỗi tác vụ chỉ thực hiện trên một vùng; hoặc theo cyclic – chu kỳ, nghĩa là mỗi tác vụ thực hiện luân phiên trên các khối, minh họa trên hình 5.3 và 5.4.



Problem Data Set: Tập dữ liệu của bài toán

Hình 5.3. Dữ liệu được chia



1D: One demension: Một chiều
2D: Two demensions: Hai chiều
Block: Khối
Cyclic: Tuần hoàn, chu trình

Hình 5.4. Các cách thực hiện của các bộ xử lý

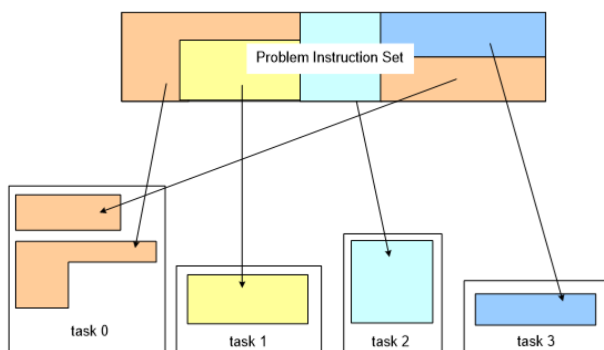
5.3.2. Phân chia theo chức năng (Domain Problem)

Cách tiếp cận này tập trung vào việc tính toán có thể thực hiện được hay không chứ không phải nói về dữ liệu được chế tác bằng cách tính toán này. Vấn đề được phân chia theo công việc mà nó phải được thực hiện. Mỗi tác vụ sau đó thực hiện một phần của công việc chung.

Ví dụ, một dây chuyền may áo sơ mi bao gồm: may thân áo, cổ áo, cửa tay, thùa khuyết, làm khuy, ... có thể chia dây chuyền này theo các công đoạn trên-mỗi công đoạn là một chức năng do các công nhân thực hiện một cách độc lập.

Khi tính diện tích tam giác theo công thức S = a*b*sin (<a,b>), có các chức năng: Nhập dữ liệu cho a > 0, b > 0, nhập góc <a,b>,

0 < <a,b> < 3.1415; các chức năng này sẽ đưa cho các tác vụ thực hiện.



Hình 5.5. Phân hoạch theo chức năng

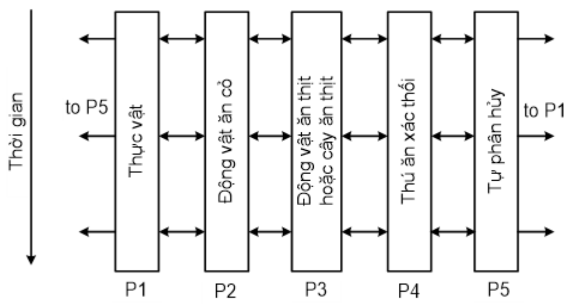
Chính các chức năng sẽ cho biết chúng có thể được chia thành các tác vụ song song hay không.

Chúng ta xét một số mô hình sau:

Mô hình hệ sinh thái (Ecosystem Modeling)

Hệ sinh thái gồm các nhóm Thực vật, Động vật ăn cỏ, Động vật ăn thịt hoặc cây ăn sâu bọ, Thú ăn xác thối, Tự phân hủy.

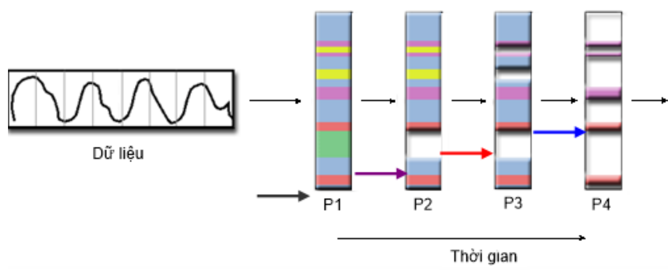
Mỗi chương trình tính toán số cá thể của một nhóm nào đó, nơi mà tăng trưởng của mỗi nhóm phụ thuộc vào sự tăng trưởng của các nhóm lân cận. Theo thời gian, mỗi quá trình sẽ tính toán trạng thái hiện thời của nó, sau đó trao đổi thông tin với nhóm lân cận. Tất cả các tác vụ sau đó tiến hành tính toán trạng thái lại cho bước tiếp theo.



Hình 5.6. Mô hình hệ sinh thái

Mô hình xử lý tín hiệu âm thanh

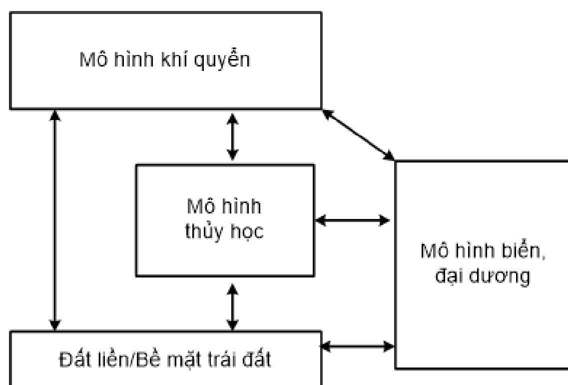
Một tập hợp dữ liệu âm thanh được truyền qua bốn bộ lọc riêng biệt có thể tính toán được. Mỗi bộ lọc là một quá trình xử lý riêng biệt. Phân đoạn đầu tiên của dữ liệu phải đi qua bộ lọc đầu tiên (bộ xử lý p1) trước khi tiến vào bộ lọc thứ hai (bộ xử lý p2). Khi p2 bắt đầu xử lý dữ liệu của phân đoạn đầu thì phân đoạn thứ hai của dữ liệu bắt đầu đi vào bộ lọc đầu tiên..., tất cả bốn bộ xử lý (p1, p2, p3, p4) làm bốn chức năng riêng biệt đều bận rộn, tức là thực hiện song song.



Hình 5.7. Mô hình xử lý tín hiệu âm thanh

Trong đó data là dữ liệu của tín hiệu, P1, P2, P3 và P4 là các bộ xử lý.

Mô hình khí hậu (Climate Modeling)



Hình 5.8. Mô hình sinh thái

Mô hình khí hậu là mô hình nói lên sự liên quan giữa các mô hình con: Mô hình khí quyển, Mô hình thủy học, Mô hình biển, đại dương, mô hình Đất liền/Bề mặt trái đất.

Mỗi thành phần mô hình có thể được dùng như một tác vụ riêng biệt. Mũi tên đại diện trao đổi dữ liệu giữa các thành phần trong quá trình tính toán: mô hình khí quyển tạo ra dữ liệu tốc độ gió được sử dụng bởi các mô hình đại dương, mô hình đại dương tạo ra dữ liệu nhiệt độ bề mặt nước biển được sử dụng bởi các mô hình khí quyển,...

Kết hợp hai loại này của phân chia vấn đề là hoàn toàn phổ biến và tự nhiên.

5.4. TRUYỀN THÔNG

Truyền thông giữa các tác vụ phụ thuộc vào vấn đề của chúng ta đang xem xét để lập trình song song.

Vấn đề có thể không cần truyền thông giữa các tác vụ

- Một số loại vấn đề có thể được tách ra thành các vấn đề con và thực hiện song song mà khi thực hiện hầu như không cần chia sẻ dữ liệu cho các tác vụ còn lại. Ví dụ, hãy tưởng tượng một thao tác xử lý ảnh, mỗi điểm ảnh trong một ảnh màu đen/trắng cần đảo màu của nó. Các dữ liệu hình ảnh có thể dễ dàng được phân phối cho nhiều tác vụ, sau đó các tác vụ hành động một cách độc lập để làm phần công việc của chúng.

- Những loại vấn đề song song có yêu cầu truyền thông liên tác vụ, nói chung là ít.

Vấn đề có thể rất cần truyền thông

Hầu hết các ứng dụng song song là giải quyết "một" vấn đề bằng cách thực hiện các tác vụ một cách đồng thời và vì thế mà các tác vụ rất cần chia sẻ dữ liệu với nhau. Ví dụ, phương trình truyền nhiệt 3 chiều, yêu cầu một tác vụ cần biết được nhiệt độ đã được tính bởi các tác vụ có dữ liệu lân cận. Thay đổi dữ liệu lân cận có tác động trực tiếp trên dữ liệu của tác vụ.

Các yếu tố cần quan tâm về truyền thông

Có một số yếu tố quan trọng cần xem xét khi thiết kế chương trình song song là vấn đề truyền thông liên tác vụ của chương trình, đó là:

Chi phí truyền thông:

- Truyền thông liên tác vụ hầu như luôn luôn mang ý nghĩa là chi phí (không phải thời gian thì là tiền bạc).

- Các thiết bị và tài nguyên có thể được sử dụng để tính toán thay vì dùng để đóng gói và truyền tải dữ liệu.

- Truyền thông, nói chung, thường xuyên đòi hỏi một số loại đồng bộ giữa các tác vụ, điều này có thể dẫn đến việc chi phí thời gian "chờ đợi" thay vì làm việc, là cái mà một chương trình song song cần giảm tối đa.

- Cạnh tranh giữa các thiết bị cũng làm ảnh hưởng lớn đến

khả năng thông lượng của băng thông mạng có sẵn và ảnh hưởng đến hiệu năng tính toán song song.

Trễ so với băng thông

- Độ trễ là thời gian cần để gửi một thông báo tối thiểu từ điểm A tới điểm B.

- Thường được diễn tả bởi đơn vị μs (1μs = một phần triệu giây).

- Băng thông là lượng dữ liệu có thể được truyền trên một đơn vị thời gian. Thường được diễn tả như MB/s (megabytes/giây) hoặc GB/s (gigabyte/giây).

- Khi gửi nhiều thông báo nhỏ có thể gây ra độ trễ làm tăng chi phí truyền thông.

- Thông thường nên đóng gói các thông báo nhỏ thành một thông điệp lớn hơn kiểu khung (frame) để làm tăng hiệu quả của các băng thông.

Tầm nhìn của truyền thông

- Với mô hình truyền thông báo, truyền thông là rõ ràng và nói chung khá hữu hình và dưới sự kiểm soát của lập trình viên.

- Với mô hình dữ liệu song song, truyền thông thường phải rất tường minh để lập trình, đặc biệt là trên những kiến trúc bộ nhớ phân phối. Lập trình viên có thể, thậm chí không biết chính xác cách thức liên lạc giữa các tác vụ đang được thực hiện như thế nào – nghĩa là vô hình với các lập trình viên.

Truyền thông đồng bộ và không đồng bộ

- Truyền đồng bộ yêu cầu một số loại thiết lập quan hệ “tay bắt tay” ("handshaking") giữa các tác vụ có chia sẻ dữ liệu. Điều này có thể được cấu trúc trong mã bởi lập trình viên, hay có thể xảy ra ở mức thấp mà các lập trình viên không biết.

- Truyền đồng bộ thường được xem là truyền và nhận trọn vẹn cả khối thông tin, do đó các tác vụ khác phải đợi cho đến khi

truyền thông đã hoàn thành.

- Truyền thông không đồng bộ cho phép các tác vụ chuyển dữ liệu một cách độc lập với nhau. Ví dụ, tác vụ X có thể gửi thông báo đến tác vụ Y, và ngay lập tức sau khi gửi nó bắt đầu thực hiện công việc khác của nó mà không quan tâm khi nào tác vụ Y thực sự nhận dữ liệu. Truyền thông không đồng bộ thường được gọi là truyền thông phi khối do công việc khác vẫn có thể được thực hiện trong khi truyền thông đang diễn ra.

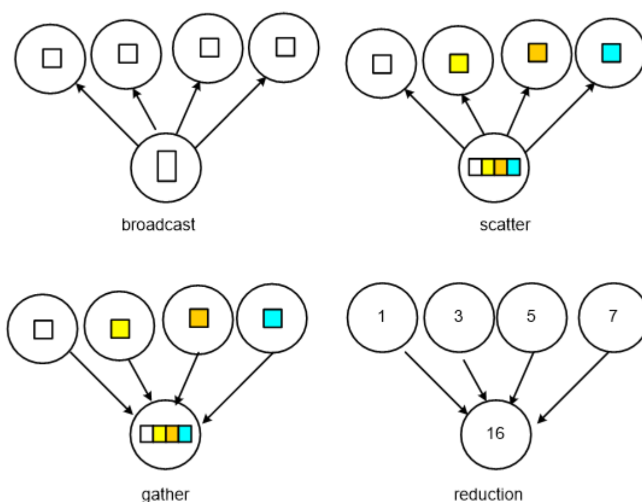
- Việc xen kẽ (Interleaving) tính toán với truyền thông tốt nhất trong trường hợp sử dụng truyền thông không đồng bộ.

Phạm vi truyền thông

- Biết được những tác vụ nào phải truyền thông với nhau là rất quan trọng trong giai đoạn thiết kế mã hóa song song. Cả hai phạm vi mô tả dưới đây có thể được triển khai thực hiện đồng bộ hoặc không đồng bộ.

- Point-to-point, liên quan đến hai tác vụ trong đó một tác vụ gửi (nơi sinh ra dữ liệu hoặc thông báo), và tác vụ còn lại đóng vai trò nhận (nơi nhận dữ liệu hoặc thông báo).

- Tập thể, bao gồm việc chia sẻ dữ liệu giữa nhiều hơn hai tác vụ, mà thường được quy định như các thành viên trong một nhóm chung, hoặc tập thể. Một số biến thể phổ biến của truyền thông được thể hiện trên hình 5.9:



Broadcast: Chương trình phát thanh hoặc truyền hình
Scatter: Phân tán
Gather: Thu, tập trung lại
Reduction: Thu gọn lại, nén

Hình 5.9. Các phạm vi truyền thông

Hiệu quả của truyền thông

- Thông thường, các lập trình viên phải lựa chọn các yếu tố có thể ảnh hưởng đến hiệu suất truyền thông.

- Thực hiện kiểu truyền thông nào cho mô hình song song đã cho? Sử dụng mô hình thông báo là một ví dụ, thực hiện thông báo trên một phần cứng cơ sở có thể nhanh hơn trên những phần cứng khác.

- Những loại thao tác truyền thông nào được sử dụng? Như đã đề cập trước đây, hoạt động truyền thông không đồng bộ có thể cải thiện hiệu suất tổng thể của chương trình.

- Mạng truyền thông hay một số mạng truyền thông cơ sở có thể cung cấp việc truyền cho một số mạng máy tính dùng đồng thời (kiểu thuê bao,...). Nhưng cái nào là tốt nhất thì cần phải được lựa chọn dựa vào mục tiêu và các ràng buộc hiện hữu.

5.5. ĐỒNG BỘ HÓA

Đồng bộ các tác vụ truyền thông

Khi thực hiện truyền thông, sẽ có một số phương thức cần phải phối hợp với các tác vụ tham gia truyền thông. Ví dụ, trước khi một tác vụ có thể thực hiện gửi, nó phải nhận được một tín hiệu sẵn sàng nhận từ các tác vụ tiếp nhận khi đó việc chuyển thông báo mới được thực hiện.

Các từ khóa khi thực hiện đồng bộ hóa:

- *Ngưỡng (Barrier)*. Ngưỡng tác vụ là khoảng thời gian để một tác vụ kết thúc công việc của mình. Ngưỡng chung là khoảng thời gian ít nhất để tất cả các tác vụ song song đều kết thúc công việc của mình.

- Thông thường, tất cả các tác vụ song song đều có ngưỡng.

- Mỗi tác vụ thực hiện công việc của mình cho đến khi nó đạt tới ngưỡng. Sau đó nó dừng lại, hay bị "chặn lại".

- Khi công việc cuối cùng đạt tới ngưỡng (ngưỡng chung), tất cả các tác vụ được đồng bộ hóa.

- Thông thường, sau đồng bộ hóa (tức là các tác vụ song song hiện thời đã kết thúc), phần tiếp theo của công việc phải được thực hiện thì hoặc là tuần tự hoặc là các tác vụ song song tiếp theo phải được tự động phát động để tiếp tục công việc. Quá trình này tiếp tục cho đến khi kết thúc công việc cần làm của chương trình.

Khóa/Cờ hiệu (Lock/semaphore)

Khóa/Cờ hiệu được sử dụng để cô lập hóa dữ liệu hay một đoạn mã nào đó. Khóa hoặc cờ hiệu sẽ được dùng:

- Cho bất kỳ tác vụ nào trong chương trình.

- Để cấp quyền truy cập vào dữ liệu toàn cục hay một phần của mã lệnh chương trình. Chỉ có một tác vụ tại một khoảng thời gian nào đó được sử dụng (riêng một mình) khóa hoặc cờ hiệu.

- Một tác vụ giành được "tập hợp" khóa hoặc cờ hiệu mới có thể truy cập vào vùng dữ liệu hoặc một đoạn mã được bảo vệ .

- Các tác vụ khác có thể cố gắng để giành các khóa hoặc cờ hiệu nhưng phải đợi cho đến khi tác vụ đang sở hữu trả lại khóa hoặc cờ hiệu.

- …

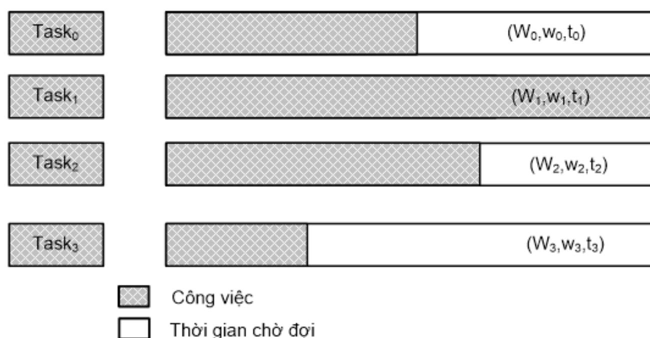
5.6. CÂN BẰNG TẢI

Cân bằng tải có liên quan đến việc phân chia công việc cho các tác vụ sao cho tất cả các tác vụ đều phải làm việc đồng thời và thời gian kết thúc của từng tác vụ là xấp xỉ nhau. Có thể coi cân bằng tải là việc giảm thiểu tổng thời gian nhàn rỗi của các tác vụ - thời gian chờ đợi của các tác vụ xong sớm hơn.

Cân bằng tải rất quan trọng cho các chương trình song song khi thực thi. Ví dụ, nếu tất cả các tác vụ không được đồng bộ hóa thì chính những công việc chậm nhất (có ngưỡng lớn nhất) sẽ ảnh hưởng đến hiệu suất tổng thể, hình 5.10. Làm thế nào để đạt được cân bằng tải? Vấn đề có thể được giải quyết bằng phương pháp chia đều hoặc gán động công việc sẽ được trình bày sau đây.

5.6.1. Phương pháp chia đều công việc cho các tác vụ

Trong trường hợp công việc được chia đều cho các tác vụ (mỗi tác vụ có thể thực hiện trên một bộ xử lý), tuy nhiên ngưỡng của mỗi tác vụ thường không giống nhau. Ví dụ, có bốn tác vụ Task0, Task1, Task2, Task3, nhận các khối lượng công việc như nhau W0 = W1 = W2 = W3, tuy nhiên thời gian kết thúc t0, t1, t2, t3 sẽ không như nhau, vì vậy sinh ra thời gian chờ đợi w0, w1, w2, w3.



Hình 5.10. Các tải của các tác vụ

Vấn đề là cần giảm tổng thời gian chờ đợi.

Có thể thấy ngay rằng:

- Đối với các thao tác mảng mỗi tác vụ thực hiện công việc giống nhau, phân tán đều dữ liệu cho các tác vụ.

- Đối với các vòng lặp, các công việc giành cho mỗi vòng lặp phải là như nhau, phân tán đều các lần lặp cho các tác vụ.

- Nếu có sự hòa nhập hỗn hợp các máy với các đặc trưng hiệu năng khác nhau đang được sử dụng thì cần đảm bảo phải dùng công cụ phân tích tốt nhất để phát hiện sự không cân bằng tải nào đó và điều chỉnh công việc cho phù hợp.

5.6.2. Phương pháp gán động công việc

- Có tồn tại một số vấn đề làm nảy sinh hiện tượng không cân bằng tải mặc dù dữ liệu đã được phân tán đều cho các tác vụ để xử lý trên các bộ xử lý như nhau.

- Những mảng thưa – một số tác vụ sẽ có dữ liệu thực để làm việc trong khi các tác vụ khác hầu như không có (mostly "zeros")

- Các phương pháp lưới tự thích nghi – một số tác vụ có thể cần phải làm tinh mạng lưới trong khi các tác vụ khác thì không cần.

- Các mô phỏng

+ Mô phỏng một số bộ phận có thể di chuyển đến hoặc từ tên miền tác vụ gốc của chúng đến miền của tác vụ khác; nơi mà các bộ phận thuộc sở hữu một số tác vụ yêu cầu làm việc nhiều hơn những tác vụ khác.

+ Khi số lượng công việc của mỗi tác vụ sẽ thực hiện được thay đổi một cách hữu ý, hoặc là không thể dự đoán, thì có thể sử dụng một bộ lập lịch - tiếp cận các tác vụ. Khi mỗi tác vụ kết thúc công việc, nó xếp hàng để nhận phần công việc mới.

+ Có thể phải thiết kế một thuật toán nhằm phát hiện và xử lý sự mất cân bằng tải.

5.7. SO SÁNH HIỆU QUẢ GIỮA TÍNH TOÁN VÀ TRUYỀN THÔNG

Thuật ngữ:

Granularity – Lõi (hay còn gọi là Hạt)

- Lõi là một thước đo chất lượng của các tỷ lệ tính toán với truyền thông. Lõi được kí hiệu là L và được xác định như sau:

L = Thời gian tính toán

Thời gian truyền thông

- Thời gian tính toán thường được tách ra khỏi thời gian truyền thông bởi các sự kiện đồng bộ hóa.

Song song mịn (Fine-grain Parallelism)

Số lượng tính toán được thực hiện giữa các sự kiện truyền thông tương đối nhỏ. Nghĩa là L nhỏ, dễ cân bằng tải. Truyền thông trên không (dùng mạng truyền thông) cao và làm giảm khả năng nâng cao hiệu năng.

Nếu L rất nhỏ, tức là quá tốt, có thể là do những yêu cầu truyền thông trên không cao và việc đồng bộ giữa các tác vụ sẽ dài hơn tính toán.

Song song thô (Coarse-grain Parallelism)

Số lượng tính toán được thực hiện giữa các sự kiện truyền thông tương đối lớn. Nghĩa là L lớn, tạo cơ hội nâng cao hiệu năng, khó cân bằng tải để có hiệu quả.

So sánh giữa song song mịn và song song thô

- Lõi hiệu quả phụ thuộc vào thuật toán và môi trường phần cứng trong đó nó thực thi.

- Trong phần lớn trường hợp chi phí truyền thông trên không liên quan đến truyền thông và đồng bộ hóa là cao so với tốc độ thực hiện, như vậy song song thô sẽ thuận lợi hơn.

- Song song mịn có thể giúp giảm chi phí trên không.

5.8. VAI TRÒ CỦA I/O

Những điều bất lợi của I/O

- Vận hành (phần cơ) của I/O nói chung là làm hạn chế việc xử lý của máy tính đặc biệt là môi trường song song.

- Các hệ thống I/O song song có thể chưa đủ tốt hoặc không có sẵn cho tất cả các nền tảng cơ bản cho xử lý song song.

- Trong một môi trường nơi mà tất cả các tác vụ dùng chung không gian tập tin, thao tác ghi có thể dẫn đến tập tin bị ghi đè.

- Thao tác đọc có thể bị ảnh hưởng bởi khả năng các máy chủ xử lý yêu cầu đọc nhiều lần cùng một lúc.

- Các thao tác I/O phải được tiến hành qua mạng (NFS, non-local) có thể gây tắc nghẽn nghiêm trọng và thậm chí làm cho các máy chủ file bị hư hỏng.

Những điều thuận lợi của I/O - Các hệ thống tập tin song song đã có sẵn để dùng như:.

- GPFS: General Parallel File System for AIX (IBM)

- Lustre: for Linux clusters (SUN Microsystems)

- PVFS/PVFS2: Parallel Virtual File System for Linux clusters (Clemson/Argonne/Ohio State/others)

- PanFS: Panasas ActiveScale File System for Linux clusters (Panasas, Inc.)

- HP SFS: HP StorageWorks Scalable File Share. Lustre based parallel file system (Global File System for Linux) product from HP

- Kỹ thuật lập trình giao diện song song I/O cho máy MPI đã có từ năm 1996 như một phần của MPI-2. Hiện nay sẵn có và miễn phí.

Cần quan tâm nghiên cứu:

- Nếu chúng ta có quyền truy cập vào một hệ thống tập tin lập trình song song, hãy nghiên cứu cách sử dụng nó.

- Giảm tổng thể I/O càng nhiều càng tốt

- Hãy hạn chế việc I/O chia nhỏ công việc, và sau đó sử dụng giao tiếp song song để phân phối dữ liệu cho các tác vụ song song. Ví dụ, tác vụ 1 có thể đọc một tập tin đầu vào và sau đó truyền dữ liệu cần thiết cho các tác vụ khác. Tương tự như vậy, tác vụ 1 có thể thao tác ghi sau khi nhận được dữ liệu cần thiết từ tất cả các tác vụ khác.

- Đối với các hệ thống bộ nhớ phân tán cùng chia sẻ không gian tập tin, hãy thực hiện I/O trong không gian tập tin địa phương, không được chia sẻ. Ví dụ, mỗi bộ xử lý có thể có không gian tập tin mà nó có thể được sử dụng. Điều này thường có hiệu quả hơn nhiều so với khi thực hiện I/O qua mạng của một thư mục chính.

- Tạo tên tập tin duy nhất cho tập tin I/O cho từng tác vụ.

Giới hạn và chi phí của lập trình song song

Luật Amdahl:

- Luật Amdahl phát biểu rằng khả năng tăng tốc chương trình được xác định bởi phần mã P được song song (đơn vị của P là %)

$$Speedup = \frac{1}{1 - P}$$

- Nếu không có phần mã được song song, P = 0 thì tăng tốc = 1 (không tăng tốc).

- Nếu tất cả mã được song song, P = 100% thì tăng tốc là vô hạn (lý thuyết).

- Nếu 50% số mã có thể được song song, tăng tốc tối đa = 2, có nghĩa là đoạn mã sẽ chạy nhanh gấp hai lần.

- Khi đưa vào một nhóm các bộ xử lý thực hiện song song các phần của công việc được chia ra, mối quan hệ có thể được mô phỏng theo:

$$Speedup = \frac{1}{P/N + S}$$

Trong đó P là phần mã được thực hiện song song, N là số của bộ vi xử lý và S là phần mã được thực hiện tuần tự, đơn vị của S là %.

Ví dụ:

Speedup			
N	P = 50%	P = 90%	P = 99%
10	1.82	5.26	9.17
100	1.98	9.17	50.25
1000	1.99	9.91	90.99
10000	1.99	9.91	99.02
100000	1.99	9.99	99.90

- Tuy nhiên, một số bài toán chỉ ra rằng hiệu năng tăng khi tăng kích thước bài toán.

Ví dụ:

- Tính toán lưới 2D: 85 giây 85%

- Phần tuần tự: 15 giây 15%

Chúng ta có thể tăng kích thước vấn đề bằng cách tăng gấp đôi kích thước lưới và giảm một nửa bước thời gian. Điều này dẫn đến bốn lần số điểm lưới và hai lần số lượng các bước thời gian. Các thời gian tính toán sau đó như sau:

- Tính toán lưới 2D: 680 giây 97,84% (2D tính toán lưới điện 680 giây 97,84%)

- Phần tuần tự: 15 giây 2,16%

- Vấn đề là tăng tỷ lệ thời gian song song với kích thước của chúng có nhiều khả năng mở rộng hơn là các vấn đề với tỷ lệ về thời gian song song cố định.

Tính phức tạp

- Nói chung, các ứng dụng song song, phức tạp hơn nhiều so với các ứng dụng tương ứng nhưng tuần tự. Vì song song hóa không chỉ phải tạo ra nhiều luồng chỉ lệnh thực hiện cùng một lúc, mà còn có sự trao đổi dữ liệu giữa chúng.